Network Decoupling: A Methodology for Secure Communications in Wireless Sensor Networks

Wenjun Gu, Xiaole Bai, Sriram Chellappan and Dong Xuan

Abstract

Many wireless sensor network (WSN) applications demand secure communications. The random key pre-distribution (RKP) protocol has been well accepted in achieving secure communications in WSNs. A host of key management protocols have been proposed based on the RKP protocol. However, due to the randomness in key distribution and strong constraint in key path construction, the RKP based protocols can only be applied in highly dense networks, which are not always feasible in practice. In this paper, we propose a methodology called *network decoupling* to address this problem. With this methodology, a wireless sensor network is decoupled into a *logical key-sharing network* and a *physical neighborhood network*, which significantly releases the constraint in key path construction of RKP protocol. We design two new key management protocols, i.e., RKP-DE and RKP-DEA, as well as a set of link and path dependency elimination rules in decoupled sensor networks. Our analytical and simulation data demonstrate the performance enhancement of our solutions from the perspective of connectivity and resilience, and its applicability in non-highly dense sensor networks.

Index Terms

Wireless Sensor Networks, Random Key Pre-distribution, Network Decoupling.

I. Introduction

In this paper, we address the issue of providing secure communications in Wireless Sensor Networks (WSNs). WSNs are gaining wide acceptance today with a host of new applications being realized involving many tiny wireless sensors performing sensing and communication tasks. Many of these applications are in hostile/vulnerable environments, and their success is contingent on preventing the WSNs information from being accessible to external malicious attackers.

Wenjun Gu, Xiaole Bai, Sriram Chellappan and Dong Xuan are with The Department of Computer Science and Engineering, The Ohio State University, Columbus, OH 43210, U.S.A. E-mail: {gu, baixia, chellapp, xuan}@cse.ohio-state.edu. An earlier version was accepted by the fourteenth IEEE International Workshop on Quality of Service (IWQoS) 2006.

This work was partially supported by NSF under grants No. ACI-0329155 and CCF-0546668. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of NSF.

Motivation: In order to provide secure communications in WSNs, secret keys need to be established between communicating sensors. A host of key distribution techniques have been proposed for wired and wireless ad hoc networks. However, they cannot be applied in WSNs due to the unique characteristics of WSNs like hostile zone deployment, ease of node capture, physical constraints in energy and memory, etc. For instance, public key cryptography [1], [2] is too energy consuming for energy constrained sensors. The key distribution center based scheme [3] is centralized and not scalable when network size increases. Using a single master key for all communications is too vulnerable, while establishing a unique pair-wise key for each pair of nodes requires too much memory. Also when many sensors are to be deployed, random deployment may be the only choice. Since neighborhood information cannot be known in advance in such cases, pre-distributing pair-wise keys between neighboring sensors prior to deployment is not possible too.

In order to address the above concerns, *Random Key Pre-distribution* (RKP) protocol was first proposed in [4]. In RKP protocol, each sensor is pre-distributed with k distinct keys randomly chosen from a key pool of K keys before deployment. Using the pre-distributed keys, two neighboring sensors attempt to establish a pair-wise key for secure communications between themselves. We denote the neighbors of a sensor with which it can establish pair-wise keys as *secure neighbors*. The RKP protocol has been well accepted in WSNs due to its good performance, distributed nature, simplicity, energy efficiency and scalability. As such, it has served as a foundation for a host of key management protocols in WSNs that aim towards improving probability of pair-wise key establishment, enhancing resilience under attack, or decreasing storage overhead [5], [6], [7], [8], [9], [10], etc.

However, most RKP based protocols have an inherent limitation in that the security performance is satisfactory only in highly dense networks, where the average number of physical neighbors per node (i.e., average physical node degree) is large ($>= 20$) [4], [5], [6]. As we know, such a high density is not always feasible in practice due to high deployment cost, increased chance of collisions, low per node throughput etc. Due to the randomness in key distribution and strong constraint in key path construction of RKP protocols (discussed subsequently), it often happens that the secure node degree (i.e., number of secure neighbors for a sensor) is very low in non-highly dense networks. Consequently, the networks will have low secure connectivity and are very likely to be partitioned as illustrated in Fig. 1. The original sensor network is shown in Fig. 1 (a). There is an edge between two nodes if they are physical neighbors. The average physical node degree is 9.71. The corresponding secure network as a result of executing

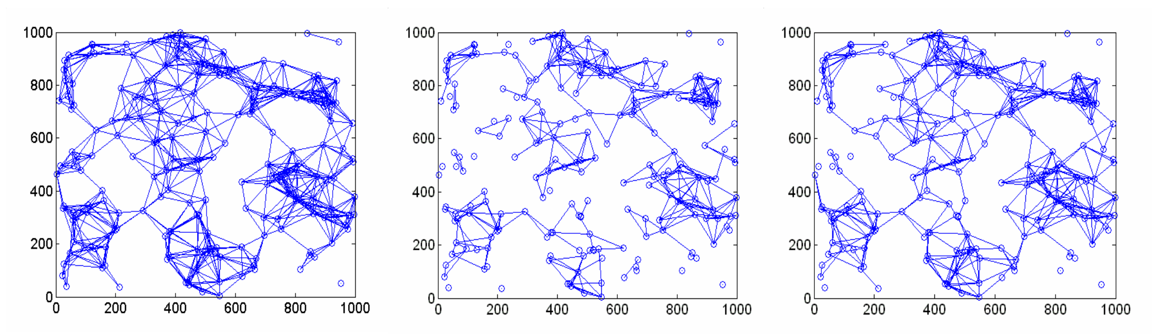


Fig. 1. Average secure node degree comparison between RKP and RKP-DE. Our RKP-DE achieves 40% improvement in average secure node degree. The network is of size $1000m$ * $1000m$, where 200 nodes are deployed uniformly at random. All nodes have the same communication range ($133m$) and the average physical node degree is 9.71. We set $K = 10000$ and $k = 50$.

the RKP protocol is shown in Fig. 1 (b) where an edge exists between two nodes if they are secure neighbors. In this example, we limit the number of intermediate nodes on a key path to be one [1]. The average secure node degree in Fig. 1 (b) is only 4.06. It is much smaller compared to the average physical node degree. As can be seen, the network in Fig. 1 (b) is partitioned into many components. Two nodes cannot communicate securely if they reside in different components.

Our Contributions: In this paper, we address the above problem. Our contributions are four-fold.

- *Network Decoupling Methodology:* We propose a methodology called *network decoupling* for secure communications in WSNs. In network decoupling, we decouple the logical key sharing relationship from the physical neighborhood relationship in sensor networks. The flexibility offered by decoupling greatly enhances the chances of pair-wise key establishment between physical neighbors.
- *Protocol Design:* We design two new key management protocols for secure neighbor establishment in decoupled sensor networks. In our first protocol called RKP-DE, logical key paths are constructed based on logical key sharing relationship, and then corresponding physical key paths are constructed based on physical neighborhood relationship. We then design an adaptive protocol called RKP-DEA, which takes network heterogeneity into consideration during protocol execution. This protocol is particularly well suited for achieving high security performance in heterogeneous sensor networks.
- *Dependency Elimination Rules:* We propose novel dependency elimination rules in our protocols to detect and eliminate link and path dependencies without compromising resilience. In pair-wise key establishment, when multiple key paths are constructed, some links (or paths) could be dependent on other links (or paths). Such dependencies introduce unnecessary overhead in terms of communication

[1]The general case of multiple intermediate nodes on a key path will be discussed later.

and computation, which can be eliminated using our proposed rules.

- *Analysis:* We conduct a formal analysis of our protocols from the perspective of average secure node degree and overhead. Our analysis demonstrates significant improvement in average secure node degree in our protocols compared to RKP protocol. We also define and analyze a new metric called $stretch\ factor$ to quantify communication overhead. Formally, stretch factor is the average number of physical hops on the key path between two secure neighbors. The stretch factor of our protocols is only slightly larger than that of RKP protocol, further demonstrating the efficiency of network decoupling. Computation overhead is studied by analyzing algorithm complexity of our protocols.

To illustrate performance improvement of network decoupling methodology, we show the corresponding secure network as a result of executing RKP-DE protocol in Fig. 1 (c) (under same configuration). The average secure node degree in Fig. 1 (c) has now increased to 5.68, a 40% improvement over that in Fig. 1 (b). Extensive analysis and simulations conducted in this paper further validate this fact.

We point out that we have done prior work on network decoupling for key management in [11]. In [11], we focus on only one intermediate node on a key path between secure neighbors. In this paper, we generalize our analysis of average secure node degree to consider arbitrary number of intermediate nodes on a key path. The analysis of the general case is much more involved. Also our work in [11] did not consider network heterogeneity. In this paper, we have proposed a new adaptive protocol $RKP-DEA$ that takes network heterogeneity into account during pair-wise key establishment, which not only helps achieve high security performance in heterogeneous networks, but also helps reduce overhead in some situations. Furthermore, we defined a new metric called $stretch\ factor$ to quantify protocol communication overhead, and we study algorithm complexity to evaluate protocol computation overhead in this paper, which were not there in [11]. Our analysis shows that our proposed protocols have reasonable and tolerable overhead for resource constrained sensor nodes. Besides, in this paper we study the tradeoff among connectivity, resilience and storage overhead with respect to key chain size (*k*) and key pool size (*K*) in detail.

The rest of our paper is organized as follows. We discuss traditional RKP protocol in Section II. The methodology of network decoupling is introduced in Section III. We discuss our basic RKP-DE protocol and adaptive RKP-DEA protocol in Sections IV and V respectively. We then present analysis and performance evaluations in Sections VI and VII respectively. After discussing related work in Section VIII, we finally conclude our paper in Section IX.

II. The Random Key Pre-distribution Protocol in Wireless Sensor Networks

In this section, we give a brief overview of the random key pre-distribution (RKP) protocol. There are two stages in this protocol [4]: *key pre-distribution* and *secure neighbor establishment*. In the *key pre-distribution* stage, each node is pre-distributed with k distinct keys randomly chosen from a large key pool of K keys, and then the nodes are deployed randomly in the network. The set of keys pre-distributed in a node is called its *key chain*. Consider an example in Fig. 2, where five nodes are deployed, and $k = 3$ and $K = 9$. The keys pre-distributed in each node are shown beside the corresponding node in braces. A solid line exists between two nodes if they are physical neighbors (within communication range), and a dashed line exists between two nodes if they are logical neighbors (share at least one key).

Once nodes are deployed, the *secure neighbor establishment* stage follows. First each node sends a message to its physical neighbors, containing its node ID and the key IDs of its pre-distributed keys. Then two physically neighboring nodes attempt to establish a pair-wise key via existing secure communication between them. Here secure communication is defined as the communication between two nodes where all messages transmitted (possibly via multi-hops) are encrypted. If two physically neighboring nodes share at least one pre-distributed key, they can establish a pair-wise key directly. In Fig. 2, nodes a and b share a key k_1. Hence node a can send the randomly generated pair-wise key to node b by encrypting it with the key k_1. Otherwise, two physically neighboring nodes may still be able to establish a pair-wise key via the help of other nodes called *proxies*. Here, a key path is attempted to be constructed comprising of one or multiple proxies, where any two successive nodes on the key path are physical neighbors and share at least one pre-distributed key, and the pair-wise key is encrypted/decrypted in *each* hop till it reaches the destination. In Fig. 2, node a does not share any key with node c, however node b can be the proxy between nodes a and c. The pair-wise key between nodes a and c is first generated by node a, and sent to node b encrypted using key k_1. Node b will decrypt the pair-wise key, encrypt it using key k_4 and send it to node c. Finally node c decrypts the pair-wise key, and uses it to encrypt/decrypt future direct communication with node a. As we can see, on the established key paths, each hop needs to satisfy both the logical (sharing pre-distributed key) constraint and the physical (within communication range) constraint. The consequence is that the RKP protocol is applicable only in highly dense sensor networks. As discussed in the next section, we make this protocol applicable in non-highly dense sensor networks by decoupling the logical and physical constraints during key path construction.

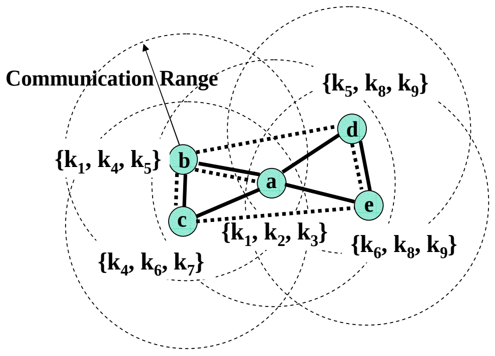


Fig. 2. Pair-wise key establishment in RKP protocol.

The standard attack model is one where the attacker attempts to decipher as much information as possible from sensor communications [4], [5], [7]. The attacker has the ability to monitor and record all the wireless communication in the network *immediately* after node deployment (i.e., link monitor attack). Besides, the attacker is assumed to be able to physically capture a limited number of nodes in the network (i.e., node capture attack). Once a node is captured, its pre-distributed keys and pair-wise keys are all disclosed to the attacker. By combining the pre-distributed keys disclosed and the messages recorded, the attacker will be able to infer the pair-wise keys between some nodes, even if the nodes themselves are not captured. For instance, if node a sends the pair-wise key to node c via node b, then this pair-wise key is inferred if either key k_1 or k_4 is disclosed (by capturing some other nodes) even though nodes a and c may not have been captured. We denote the pair-wise keys disclosed by the attacker as $compromised$.

To evaluate the performance of RKP protocol, two types of metrics are considered. The first is *connectivity*, which includes *local connectivity* and *global connectivity*. Local connectivity is defined as the probability that two physically neighboring nodes are able to establish a pair-wise key between them. Global connectivity is defined as either the probability that the whole secure network (e.g., Fig. 1 (b) or (c)) is connected, or the percent of nodes in the largest connected component of the secure network. The other performance metric is *resilience*, which is defined as the probability that a pair-wise key between two nodes is not compromised given that those two nodes are not captured. The overall goal clearly is to make connectivity and resilience as high as possible.

III. Network Decoupling in Random Key Pre-distributed Sensor Networks

A. Network Decoupling

In random key pre-distributed sensor networks, there exist two types of relationship between any two nodes. One is logical (sharing pre-distributed keys), and the other is physical (within communication

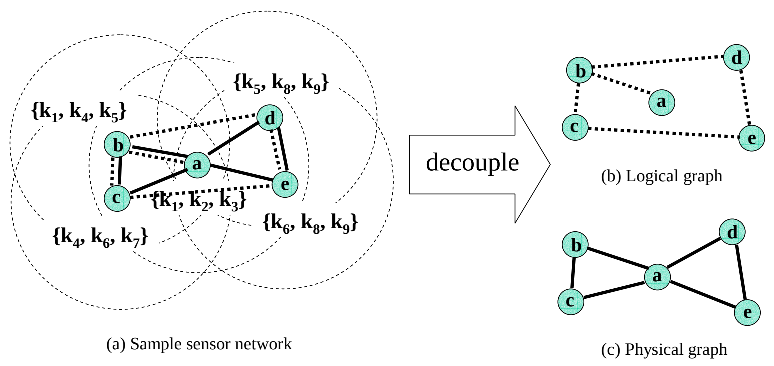


Fig. 3. Decouple a sensor network into a logical graph and a physical graph.

range). We can separate these two types of relationship by decoupling a random key pre-distributed sensor network into two graphs: a logical one and a physical one. Two nodes in the logical graph have an edge between them if they share at least one pre-distributed key. Similarly two nodes in the physical graph have an edge between them if they are within communication range of each other. For the example in Fig. 3 (a), its decoupled logical and physical graphs are shown in Fig. 3 (b) and (c) respectively. Detailed description on how nodes construct these graphs is presented in Section IV.B.

Recall that secure communication is defined as the communication between two nodes where all messages transmitted are encrypted. Now we will show how network decoupling achieves secure communication. There are two cases possible, where two nodes in the network can communicate securely. The first case is where two nodes share at least one pre-distributed key (i.e., they are *directly* connected in the logical graph) and the nodes are connected (via one or more hops) in the physical graph. In this case, the source node can encrypt the messages using the shared key, and each intermediate node in the physical graph simply forwards the messages towards the destination. Such intermediate nodes in the physical graph are called as *physical intermediate nodes*. An example of the first case is nodes b and d in Fig. 3 (a), where node a is the physical intermediate node between them. The second case is one where two nodes do not share a key (i.e, they are *not directly* connected in the logical graph), but are connected *indirectly* in the logical graph with multiple logical hops, and the two nodes for each logical hop are connected (directly or indirectly) in the physical graph. In this case, encryption occurs at each intermediate node in the logical graph, while each intermediate node in the physical graph simply forwards the messages. Such intermediate nodes in the logical graph are called as *logical intermediate nodes*. An example of the second case is nodes a and d in Fig. 3 (a), where node b is the logical intermediate node between them. Note that if a physical (or logical) intermediate node also satisfies the logical (or physical) constraint, such node becomes a $proxy$ as defined in the RKP protocol (in the coupled network).

B. Benefit of Networking Decoupling

In this section, we will demonstrate the benefits of network decoupling by analysis. Specifically, we will derive the probability that two physically neighboring nodes are able to establish a pair-wise key. In our previous work in [11], we present the analysis for the simple case where one proxy or logical intermediate node is used on a key path. In this work, we extend the analysis to the general case where arbitrary number of proxies or logical intermediate nodes are used on a key path. Our analysis here will be used later in the analysis of average secure node degree in Section VI.

Formally, the probabilities that two physically neighboring nodes are able to establish a pair-wise key in the RKP protocol (with multiple proxies), denoted by P_{couple}, and the RKP-DE protocol (with multiple logical intermediate nodes), denoted by $P_{decouple}$, are given by,

$$P_{couple} = 1 - (1 - P_{couple}(A_a)) \cdot (1 - P_{couple}(A_b|\bar{A}_a)), \tag{1}$$

$$P_{decouple} = 1 - (1 - P_{decouple}(A_a)) \cdot (1 - P_{decouple}(A_b|\bar{A}_a)). \tag{2}$$

In the above expressions, $P_{couple}(A_a)$ and $P_{decouple}(A_a)$ denote the probability that node a is able to construct a key path to its physical neighbor node b based on local information within its information area in the RKP and RKP-DE protocols respectively. By default, the information area is the communication range with size $A_a = \pi r^2$, where r is the communication range. In the above, $P_{couple}(A_b|\bar{A}_a)$ and $P_{decouple}(A_b|\bar{A}_a)$ denote the probability that node b is able to construct a key path to node a based on its local information (within range $A_b = \pi r^2$), given node a cannot construct a key path to node b within A_a, in the RKP and RKP-DE protocols respectively. In the following, we will derive the expressions for $P_{decouple}(A_a)$ and $P_{decouple}(A_b|\bar{A}_a)$ first, followed by the derivation for $P_{couple}(A_a)$ and $P_{couple}(A_b|\bar{A}_a)$.

We first define some notations. We define $P_h(i)$ as the probability that a node a can find a logical key path to a physical neighbor node b in our RKP-DE protocol within the information area of node a with minimum logical hops i. Similarly $P_h(i|A')$ is defined as the probability that node a can find a logical key path to node b in RKP-DE protocol within the overlapped information area of nodes a and b with minimum logical hops i. We define $P_{decouple}(A')$ as the probability that node a is able to find a key path to node b within the overlapped information areas of nodes a and b ($A' = 0.5865 \cdot \pi r^2$[5]), and define $P_{decouple}(\bar{A}')$ as $1 - P_{decouple}(A')$. $P_{decouple}(A_a|A')$ and $P_{decouple}(A_a|\bar{A}')$ are conditional probabilities defined similarly as $P_{decouple}(A_b|\bar{A}_a)$ above. By the law of total probability, we have,

$$P_{decouple}(A_a) = P_{decouple}(A') \cdot P_{decouple}(A_a|A') + P_{decouple}(\bar{A}') \cdot P_{decouple}(A_a|\bar{A}'). \quad (3)$$

In the above equation, $P_{decouple}(A_a)$ is given by $P_{decouple}(A_a) = \sum_{i=1}^{\infty} P_h(i)$. Similarly, $P_{decouple}(A')$ is given by $P_{decouple}(A') = \sum_{i=1}^{\infty} P_h(i|A')$. The value of $P_{decouple}(A_a|A')$ is always one since area A' is within area A_a. Therefore, we can obtain the expression of $P_{decouple}(A_a|\bar{A}')$ by,

$$\begin{aligned} P_{decouple}(A_a|\bar{A}') &= (P_{decouple}(A_a) - P_{decouple}(A'))/P_{decouple}(\bar{A}') \\ &= (\sum_{i=1}^{\infty} P_h(i) - \sum_{i=1}^{\infty} P_h(i|A'))/(1 - \sum_{i=1}^{\infty} P_h(i|A')). \end{aligned} \quad (4)$$

Noting that $P_{decouple}(A_b|\bar{A}_a)$ equals $P_{decouple}(A_a|\bar{A}')$. Finally, we have the expression of $P_{decouple}$,

$$\begin{aligned} P_{decouple} &= 1 - (1 - P_{decouple}(A_a)) \cdot (1 - P_{decouple}(A_b|\bar{A}_a)) \\ &= 1 - (1 - \sum_{i=1}^{\infty} P_h(i)) \cdot (1 - (\sum_{i=1}^{\infty} P_h(i) - \sum_{i=1}^{\infty} P_h(i|A'))/(1 - \sum_{i=1}^{\infty} P_h(i|A'))). \end{aligned} \quad (5)$$

The detailed derivation of $P_h(i)$ and $P_h(i|A')$ are given in Appendix.

The analysis of RKP protocol is similar to that of RKP-DE protocol above. The main difference comes from the fact that two successive nodes on the logical key path in RKP protocol have to be physical neighbors. If we denote $P'_h(i)$ as the probability that node a can find a logical key path to a physical neighbor node b in the RKP protocol within the information area of node a with minimum logical hops i, and denote $P'_h(i|A')$ as the probability that node a can find a logical key path to node b in RKP protocol within the overlapped information areas of nodes a and b with minimum logical hops i, the expression of P_{couple} is given by,

$$\begin{aligned} P_{couple} &= 1 - (1 - P_{couple}(A_a)) \cdot (1 - P_{couple}(A_b|\bar{A}_a)) \\ &= 1 - (1 - \sum_{i=1}^{\infty} P'_h(i)) \cdot (1 - (\sum_{i=1}^{\infty} P'_h(i) - \sum_{i=1}^{\infty} P'_h(i|A'))/(1 - \sum_{i=1}^{\infty} P'_h(i|A'))). \end{aligned} \quad (6)$$

Due to space limitation, we skip the derivation of $P'_h(i)$ and $P'_h(i|A')$. Interested readers are referred to [27] for details.

Based on our analysis in Appendix, we will see that $P_h(i)$ is larger than $P'_h(i)$, $P_h(i|A')$ is larger than $P'_h(i|A')$, and $P_h(i) - P_h(i|A')$ is larger than $P'_h(i) - P'_h(i|A')$ for all i larger than 1 (corresponding parameters are equal when $i = 1$). By observing expressions (5) and (6) above, we can see that $P_{decouple}$ is always larger than P_{couple}, which shows the benefit of our network decoupling.

IV. Secure Neighbor Establishment Protocol in Decoupled Networks

A. Overview

In this section, we discuss the design of our RKP-DE protocol for establishing secure neighbors (i.e., establishing pair-wise keys) in decoupled random key pre-distributed sensor networks. The protocol has four major components in its execution: 1) constructing local logical and physical graphs in the decoupled network for each node, 2) establishing multiple key paths between neighboring nodes, 3) eliminating dependencies among the key paths, and 4) establishing pair-wise keys between neighboring nodes. The major differences between our RKP-DE protocol and the traditional RKP protocol are in the first three components. In the following, we will describe each component in our RKP-DE protocol in detail.

B. Local Graphs Construction

After node deployment, each node obtains the key sharing and physical neighborhood information in its information area by local communication with its physical neighbors. With this information, each node constructs a local logical graph (G_l) and a local physical graph (G_p). In the local logical graph (e.g., Fig. 3 (b)), two nodes are connected if they share at least one key, while in the local physical graph (e.g., Fig. 3 (c)), two nodes are connected if they are within communication range of each other. In Algorithm 1, we show the pseudocode of local graphs construction executed by a node u.

In Algorithm 1, u denotes an arbitrary node, while $G_l(u)$ and $G_p(u)$ are its local logical and physical graphs respectively. Here $N(u)$ denotes the set of physical neighbors of node u. Node u only needs to send two messages to its neighbors (lines (2) and (4)), and needs to receive two messages from each of its physical neighbors (lines (3) and (5)). The overall communication overhead is proportional to the node density. The node IDs of node u's neighbors sent out in line (4) is obtained from the messages received earlier in line (3). After obtaining the messages received in line (5), node u is able to determine whether any two of its physical neighbors are physical neighbors or not. This information will be used to construct local physical graph in lines (15) and (16).

C. Key Paths Construction

Algorithm 2 shows the pseudocode of key paths construction executed by node u. Initially the logical key path tree (T_u) consists of a single vertex u. The key paths construction is executed in two steps. First, T_u is constructed by node u based on its local logical graph $G_l(u)$ (lines 1 to 7) and T_u contains all the

Algorithm 1 Pseudocode of Local Graphs Construction

```
1: Local_Graphs_Construction($u$,$G_l(u)$,$G_p(u)$)
2: $send\ a\ message\ to\ all\ physical\ neighbors,\ containing\ node\ ID\ and\ key\ IDs\ of\ the\ keys\ in\ key\ chain;$
3: $receive\ messages\ from\ physical\ neighbors,\ containing\ node\ IDs\ and\ key\ IDs\ of\ physical\ neighbors;$
4: $send\ a\ message\ to\ all\ physical\ neighbors,\ containing\ node\ IDs\ of\ node\ u's\ physical\ neighbors;$
5: $receive\ messages\ from\ physical\ neighbors,\ containing\ node\ IDs\ of\ node\ u's\ neighbors'\ neighbors;$
6: $G_l(u) = G_p(u) = \Phi;$
7: $add\ vertex\ u\ into\ G_l(u)\ and\ G_p(u);$
8: for each $v \in N(u)$
9:    $add\ vertex\ v\ into\ G_l(u)\ and\ G_p(u);$
10: endfor
11: for $any\ two\ nodes\ v,\ w \in N(u) \cup \{u\}$
12:    if $v\ and\ w\ share\ pre-distributed\ keys$, then
13:       $add\ an\ edge\ between\ v\ and\ w\ in\ G_l(u);$
14:    endif
15:    if $v\ and\ w\ are\ physical\ neighbors$, then
16:       $add\ an\ edge\ between\ v\ and\ w\ in\ G_p(u);$
17:    endif
18: endfor
```

Algorithm 2 Pseudocode of Key Paths Construction

```
1: Logical_Key_Path_Tree_Construction($u$,$G_l(u)$,$T_u$)
2: for each $v \in G_l(u)$
3:    if $Link\_Dependency\_Checking(v, u, T_u) == PASS$, then
4:       $Insert(u, v, T_u);$
5:       $Logical\_Key\_Path\_Tree\_Construction(v, G_l(u), T_u);$
6:    endif
7: endfor

8: Physical_Key_Paths_Construction($u$,$G_p(u)$,$T_u$)
9: for each $v \in N(u)$
10:    $obtain\ the\ set\ of\ all\ logical\ key\ paths\ between\ u\ and\ v\ (T_{uv})\ from\ T_u;$
11:    $T'_{uv} = Path\_Dependency\_Checking(T_{uv});$
12:    $obtain\ the\ corresponding\ set\ of\ physical\ key\ paths\ T^*_{uv}\ from\ T'_{uv};$
13: endfor

14: Insert($u$,$v$,$T_u$)
15:    $Insert\ node\ v\ into\ T_u\ as\ a\ child\ of\ node\ u.$
```

logical key paths between u and its secure neighbors. Then, node u constructs corresponding physical key paths based on both T_u and its local physical graph $G_p(u)$ (lines 8 to 13). The dependency checking in lines 3 and 11 will be discussed in the next subsection.

Logical key path tree construction: The protocol constructs logical key path tree (lines 1 to 7) using a variant of the standard depth-first-search algorithm, in which a node could be chosen multiple times (on different paths). Fig. 4 shows the resultant logical key path tree for node a in the example of Fig. 3 (b). By executing the algorithm just once on its local logical graph in Fig. 3 (b), node a is able to obtain all logical key paths to all its neighbors. Taking node e as an example, node a obtains two logical key paths between node a and node e, that are $< a, b, c, e >$ and $< a, b, d, e >$.

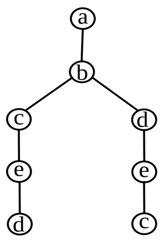


Fig. 4. Logical Key Path Tree of Node a

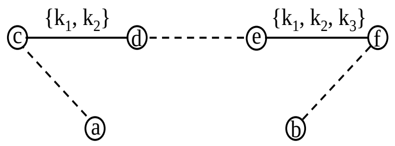


Fig. 5. Link Dependency Example

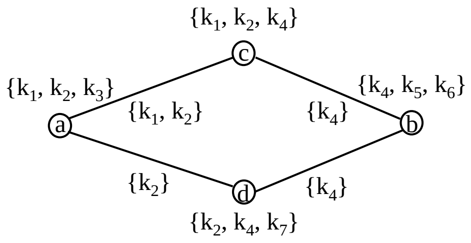


Fig. 6. Path Dependency Example

Physical key paths construction: After obtaining the logical key path tree (T_u), node u begins to construct physical key paths (lines 8 to 13). For each neighbor v, node u first obtains a set of logical key paths between u and v (T_{uv}) from T_u. Out of all paths in T_{uv}, some of them will be eliminated by dependency checking discussed later. The set of paths that pass the dependency checking is denoted as T'_{uv}. Finally, for all logical key paths in T'_{uv}, corresponding physical key paths T^*_{uv} are obtained. In Fig. 3 (b), the logical key path $< a, b, d, e >$ contains a logical hop $< b, d >$ between two non-neighboring nodes. From Fig. 3 (c), we see that a physical path $< b, a, d >$ can replace the above logical hop. Therefore, for logical key path $< a, b, d, e >$, its corresponding physical key path is $< a, b, a, d, e >$. Message encryption/decryption occurs for each logical hop, while message transmission occurs for each physical hop. Here, we select the physical path with fewest hops to replace logical hops between non-neighboring nodes. Other policies can be chosen if energy consumption, load balancing, etc. are to be considered.

D. Dependency Elimination

We now discuss elimination of link and path dependencies in steps 3 and 11 of Algorithm 2. Generally, if more key paths are used, resilience is enhanced. This is because the attacker needs to compromise all key paths in order to compromise the established pair-wise key. However, this is not always true. Existing links (or paths) may have dependencies among them such that the compromise of some links (or paths) automatically leads to the compromise of other dependent links (or paths). Clearly, the presence of such dependency does not enhance resilience. They only increase overhead in terms of both storage and energy consumption. In this subsection, we propose two novel *dependency elimination rules* to decrease such overheads without affecting the resilience of the established pair-wise keys.

1) Link Dependency Elimination: We illustrate link dependency with an example in Fig. 5. Node a obtains a logical key path $< a, \cdots, c, d, \cdots, e, f, \cdots, b >$ to a physical neighbor node b. We denote $K(i, j)$ as the set of keys shared by nodes i and j. There exists a link dependency between the hops $< c, d >$ and $< e, f >$ in that $K(c, d) \subseteq K(e, f)$. Since both nodes c and f share keys k_1 and k_2, there must

Algorithm 3 Pseudocode of Dependency Checking

```
1:  **Link_Dependency_Checking**($v$,$u$,$T$)
2:  **if** $\exists$ node $w \in$ Path($u$,$root$), s.t. $K(v, v.par) \subseteq K(w, w.par)$ OR $K(w, w.par) \subseteq K(v, v.par)$, **then**
3:    $return\ FAIL$;
4:  **else** $return\ PASS$;
5:  **endif**

6:  **Path_Dependency_Checking**($T_{uv}$)
7:  $T'_{uv} = T_{uv}$;
8:  **while** $\exists$ paths $p$ and $q \in T'_{uv}$, s.t. $p$ is weaker than $q$ OR $q$ is weaker than $p$, **do**
9:    **if** $p$ is weaker than $q$, **then**
10:     $T'_{uv} = T'_{uv} \setminus p$;
11:   **else** $T'_{uv} = T'_{uv} \setminus q$;
12:   **endif**
13: **endwhile**
14: $return\ T'_{uv}$;
```

exist another shorter logical key path $< a, \cdots, c, f, \cdots, b >$, which has better resilience than the original one. This is because the compromise of any logical hop between nodes d and e will compromise the original key path, while it is possible that the shorter key path is not compromised. On the other hand, the compromise of the shorter key path will definitely compromise the original key path. Besides, using a shorter key path will save overhead. We formally define link dependency below.

Link Dependency: Given two logical hops $< i_1, j_1 >$ and $< i_2, j_2 >$ in a logical key path, there exists link dependency between these two hops if either $K(i_1, j_1) \subseteq K(i_2, j_2)$ or $K(i_2, j_2) \subseteq K(i_1, j_1)$.

Once a link dependency is detected, our *link dependency elimination rule* will eliminate the logical key path it resides. In the above example, logical key path $< a, \cdots, c, d, \cdots, e, f, \cdots, b >$ will be eliminated. As we discussed above, the elimination of this key path will not affect the resilience of the pair-wise key. The pseudocode of link dependency checking is given in Algorithm 3 (lines 1 to 5). In Algorithm 3, $root$ denotes the root node of the logical key path tree T, $Path(u, root)$ denotes the set of nodes on the logical key path from u to $root$, and $v.par$ denotes the parent node of node v on the tree T.

2) Path Dependency Elimination: Apart from link dependency, another type called path dependency may exist. In Fig. 6, there are two logical key paths between nodes a and b. However, we can see that the compromise of the key path $< a, c, b >$ (disclosure of keys $(k_1,\ k_2)$ or (k_4)) always leads to the compromise of the other key path $< a, d, b >$, but not vice versa. Therefore, given that key path $< a, c, b >$ exists, the other key path $< a, d, b >$ becomes redundant, and also incurs unnecessary overhead. Denoting the set of logical hops on a logical key path p as L_p, and denote the set of keys used on a logical hop h as $K(h)$, path dependency is formally defined as follows.

Path Dependency: Given two logical key paths p and q, there exists path dependency between p and q if either of the following two conditions is satisfied. *(1)* $\forall\ logical\ hop\ h \in L_q,\ \exists\ a\ logical\ hop\ h'\ (h' \in L_p),\ s.t.\ K(h') \subseteq K(h)$; *(2)* $\forall\ logical\ hop\ h \in L_p,\ \exists\ a\ logical\ hop\ h'\ (h' \in L_q),\ s.t.\ K(h') \subseteq K(h)$.

If the first condition of path dependency is satisfied, we call path p $weaker$ than path q. Similarly, path q is $weaker$ than path p if the second condition is satisfied. Our *path dependency elimination rule* is that after detecting path dependency between two logical key paths, the weaker one will be eliminated. In the above example, the logical key path $< a, d, b >$ will be eliminated. In case two paths satisfy both conditions above, we can eliminate one of them based on certain policies (e.g., the path with more physical hops). The pseudocode of path dependency checking is given in Algorithm 3 (lines 6 to 14).

E. Pair-wise Key Establishment

Once key paths are constructed, each sensor generates random key shares, and sends each key share on each physical key path. The messages are transmitted at each physical hop, while are encrypted/decrypted at each logical hop. Take the logical key path $< a, b, d >$ in Fig. 3 (b) as an example. Its corresponding physical key path is $< a, b, a, d >$. We assume a key share $k_a^{(1)}$ is transmitted on this key path. We denote $\{M\}_k$ as a message M encrypted with key k. The transmission of $k_a^{(1)}$ is executed as follows:

$$\begin{aligned} a \mapsto b:&\quad <1>, \{<a>, <a, d, 5>, <k_a^{(1)}>\}_{k_1}, \\ b \mapsto a:&\quad <d>, <5>, \{<a>, <\phi>, <k_a^{(1)}>\}_{k_5}, \\ a \mapsto d:&\quad <5>, \{<a>, <\phi>, <k_a^{(1)}>\}_{k_5}. \end{aligned}$$

In the message that node a sends to node b, $< 1 >$ denotes the ID of the key used to encrypt the remaining message, $< a >$ denotes the source node, and $< a, d, 5 >$ denotes the remaining physical key path and the ID of the key used to encrypt the message forwarded to the next node d on the logical key path [2]. In the message that b sends to a, $< d >$ denotes the remaining physical path of the current logical hop since a cannot decrypt the message using key k_5.

Similarly, node a can transmit another random key share $k_a^{(2)}$ on another logical key path $< a, b, c, e, d >$ to node d. Node d may also construct other key paths (not shown in Fig. 3 (b)), and transmit its key shares to node a. Finally, nodes a and d can compute a common pair-wise key via some simple operation such as bit-wise XOR operation, based on all the key shares they both generated. In this way, the established pair-wise key is compromised if and only if all the key shares (key paths) are compromised.

[2]The next node on the logical key path is the node before the first number in the list.

V. Adaptive Secure Neighbor Establishment Protocol

A. Motivation

In our RKP-DE protocol discussed in Section IV, all nodes execute the protocol in a homogeneous way. That is, all nodes have the *same* information area (one hop communication range), and all nodes construct *all* key paths available for pair-wise key establishment similar to each other. However, in reality it may happen that despite random deployment of nodes and pre-distribute keys, there will be heterogeneity in the network. This heterogeneity can be in terms of number of physical neighbors per nodes and the chances of nodes sharing pre-distributed keys with their neighbors. In such cases, a homogeneous protocol may not achieve satisfactory performance. Let us denote *poor* nodes as those that cannot construct key paths and establish pair-wise keys with most of their neighbors even if they utilize all the key paths available, and denote *rich* nodes as those that can construct quite a few key paths to most of their neighbors. In such cases, poor nodes need to find more key paths by increasing their information areas (enhancing connectivity and resilience), while rich nodes may not need to utilize all key paths (decreasing overhead). Our adaptive RKP-DEA protocol achieves this objective.

Same as the RKP-DE protocol, our RKP-DEA protocol also has four components: local graphs construction, key paths construction, dependency elimination and pair-wise key establishment. We propose adaptive mechanisms for the first two components in our RKP-DEA protocol, which will be discussed in detail below. The other two components in our RKP-DEA protocol are the same as those in RKP-DE protocol, and will not be repeated here.

B. Adaptive Local Graphs Construction

In the RKP-DE protocol, the size of information area is the same for all nodes. As discussed above, local node density and key sharing can vary across the network. Therefore, the size of information area should be decided by each node based on its local node density and local key sharing, the details of which are discussed below. After information area size is decided, each node exchanges information with the other nodes in its information area, and constructs local logical/physical graphs based on the obtained information in the same way as it does in our RKP-DE protocol discussed in Section IV.B.

After node deployment, each node has an initial information area of its one hop communication range. Each node constructs key paths to its neighbors based on the node information and key sharing information in its initial information area. Since there is no guarantee that pair-wise key can be established between any two neighboring nodes due to the randomness of key pre-distribution, we need a threshold ($THRESH\_SN$), which denotes the required percent of neighbors with which a pair-wise key can be established (e.g., percentage of secure neighbors). The value of $THRESH\_SN$ can be decided based on the required probability that the whole secure network is connected (P_c) and the total number of nodes (N) by random graph theory in [15],

$$THRESH\_SN = \frac{\ln N - \ln(-\ln P_c)}{N}. \tag{7}$$

If a node s can achieve the above threshold, its information area will keep the same. Otherwise, node s will increase its information area to be its two-hop communication range. Besides, node s will request its current non-secure neighbors to increase their information area by one more hop as well. In this way, node s will have a higher chance of finding key paths between itself and its non-secure neighbors. Both node s and its non-secure neighbors obtain extra information in their new information areas, and construct key paths based on updated information. After this, node s will check whether the threshold is achieved or not. Depending on the result of the check, node s will decide whether to stop increasing its information area or increase it further by one more hop. In order to prevent message flooding incurred by a few poor nodes that can never achieve the above threshold, we need an upperbound of information area size $MAX\_INFOAREA$, which is the maximum hop number of information area. Having a few nodes not achieving the above threshold will not pose major impact to the global network security performance. The value of $MAX\_INFOAREA$ can be decided by our analysis in Section III. Given the information area size A, we can obtain the percentage of secure neighbors $P_{decouple}(A)$ in equation (5). The value of $MAX\_INFOAREA$ can be set as twice the minimum h such that $P_{decouple}(\pi(hr)^2) \geq THRESH\_SN$, where $THRESH\_SN$ is given in equation (7) above.

C. Adaptive Key Paths Construction

In the RKP-DE protocol, each node constructs all key paths available for pair-wise key establishment. As we point out above, establishing a pair-wise key via all the key paths could be an overkill when the number of available key paths is relatively large. In our adaptive key paths construction, we allow each node to select a subset of key paths for pair-wise key establishment. The pseudocode of our adaptive key paths construction is given in Algorithm 4.

Algorithm 4 Pseudocode of Adaptive Key Paths Construction

```
 1: Adaptive_Logical_Key_Path_Tree_Construction($u$,$G_l(u)$,$T_u$)
 2: for each $v \in G_l(u)$
 3:   if $Link\_Dependency\_Checking(v, u, T_u) == PASS$, then
 4:     $Insert(u, v, T_u)$;
 5:     $Logical\_Key\_Path\_Tree\_Construction(v, G_l(u), T_u)$;
 6:   endif
 7: endfor
 8: for each $v \in N(u)$
 9:   $obtain\ the\ set\ of\ all\ m\ logical\ key\ paths\ between\ u\ and\ v\ (T_{uv})\ from\ T_u$;
10:   $compute\ path\ resilience\ of\ all\ m\ logical\ key\ paths\ in\ T_{uv}$;
11:   $sort\ the\ m\ logical\ key\ paths\ based\ on\ path\ resilience\ (R_{path}(P_1) \geq R_{path}(P_2) \geq \cdots \geq R_{path}(P_m))$;
12:   for $(i = 1; i \leq m; i++)$
13:     $T_{uv-adaptive} = T_{uv-adaptive} \bigcup \{P_i\}$;
14:     $R_{key} = 1 - \prod_{j=1}^{i}(1 - R_{path}(P_j))$;
15:     if $R_{key} \geq THRESH\_RES$ then
16:       $break$;
17:     endif
18:   endfor
19: endfor

20: Adaptive_Physical_Key_Paths_Construction($u$,$G_p(u)$,$T_{uv-adaptive}$)
21: for each $v \in N(u)$
22:   $T'_{uv-adaptive} = Path\_Dependency\_Checking(T_{uv-adaptive})$;
23:   $obtain\ the\ corresponding\ set\ of\ physical\ key\ paths\ T^*_{uv-adaptive}\ from\ T'_{uv-adaptive}$;
24: endfor

25: Insert($u$,$v$,$T_u$)
26:   $Insert\ node\ v\ into\ T_u\ as\ a\ child\ of\ node\ u.$
```

The main difference between our adaptive key paths construction and the basic key paths construction in Algorithm 2 in Section IV is shown between lines (8) and (19) in Algorithm 4. We first compute the path resilience for all logical key paths in line (10). Then we sort the logical key paths based on path resilience in descending order in line (11). In lines (12) to (18), we add one logical key path into the selected subset of key paths $T_{uv-adaptive}$ (initially empty) in each loop until either all logical key paths are chosen or the key resilience based on currently selected subset of logical key paths (computed in line (14)) satisfies the required key resilience threshold $THRESH\_RES$. The value of $THRESH\_RES$ can be decided by the security requirement of the application. In the following, we will discuss how to derive path resilience and key resilience.

We first derive hop resilience $R_{hop}(k_i)$, which is defined as the probability that a logical hop with k_i shared keys is not compromised. The expression of $R_{hop}(k_i)$ is given by,

$$R_{hop}(k_i) = 1 - \binom{K - k_i}{k_{dis} - k_i} / \binom{K}{k_{dis}}, \tag{8}$$

where K is the key pool size, and k_{dis} is the average number of disclosed keys given x nodes are captured. This is because a logical hop with k_i shared keys is compromised if all k_i key are disclosed. The expression for k_{dis} is given by $k_{dis} = K \cdot \Big(1 - (1 - \frac{k}{K})^x\Big)$.

For a logical key path P with h logical hops, we denote the number of shared keys on each logical hop by k_i $(1 \leq i \leq h)$, and denote the path resilience of path P by $R_{path}(P)$. The expression for $R_{path}(P)$ is given by,

$$R_{path}(P) = \prod_{i=1}^{h} R_{hop}(k_i). \tag{9}$$

This is because a logical key path is uncompromised if all the logical hops are uncompromised.

For a pair-wise key established by a set of logical key paths (P_1, P_2, $\cdots$, P_m), its resilience, denoted by R_{key}, can be given by,

$$R_{key} = 1 - \prod_{i=1}^{m} (1 - R_{path}(P_i)). \tag{10}$$

This is because a pair-wise key is compromised if at least one logical key path is uncompromised.

The computation of path and key resilience above is based on the assumption that shared keys on different logical hops in the same or different key path(s) are independent. Computation of path and key resilience considering the link and path dependency is much more complicated and computationally expensive for energy constrained sensors. Our approximation above is simple but effective. Besides, in the derivation of k_{dis}, we need to know the number of captured nodes x. The exact value of x may not be known during our protocol execution, however, we can estimate x based on historical data.

VI. Analysis

In this section, we present theoretical analysis on our proposed protocols in terms of security performance and protocol overhead. We analyze security performance by deriving expressions for average secure node degree. We analyze protocol overhead by communication and computation overhead. Communication overhead is captured by *stretch factor*, while computation overhead is captured by algorithm complexity.

A. Average Secure Node Degree

In this section, we will derive the expressions for average secure node degree in both RKP and RKP-DE protocols, denoted by D_s^{RKP} and D_s^{RKP-DE} respectively. In Section III, we gave the expressions for P_{couple} and $P_{decouple}$ in equations (6) and (5) respectively, which denote the probability that two physically neighboring nodes are able to construct a key path in the RKP protocol and RKP-DE protocol respectively. Therefore, we can derive D_s^{RKP} and D_s^{RKP-DE} as,

TABLE I
COMPARISON OF AVERAGE SECURE NODE DEGREE IN RKP PROTOCOL AND RKP-DE PROTOCOL UNDER DIFFERENT D_p.

D_p	5	10	15	20	25
$D_s^{RKP(1)}$	1.66	4.26	7.59	11.52	15.89
$D_s^{RKP-DE(1)}$	2.27	6.16	10.96	16.22	21.68
$IM^{(1)}$	37%	45%	44%	41%	36%
D_s^{RKP}	1.93	6.07	11.62	17.75	23.80
D_s^{RKP-DE}	2.63	8.05	14.43	19.85	24.98
IM	37%	33%	24%	12%	5%

$$D_s^{RKP} = D_p \cdot P_{couple}, \tag{11}$$

$$D_s^{RKP-DE} = D_p \cdot P_{decouple}, \tag{12}$$

where recall that D_p denotes the average physical node degree. The improvement of D_s^{RKP-DE} over D_s^{RKP}, denoted by IM, is then given by,

$$IM = \frac{D_s^{RKP-DE} - D_s^{RKP}}{D_s^{RKP}}. \tag{13}$$

We compute the values of D_s^{RKP}, D_s^{RKP-DE} and IM under different D_p in Table I ($K = 10000$, $k = 50$). In Table I, we also give the values of $D_s^{RKP(1)}$, $D_s^{RKP-DE(1)}$ and $IM^{(1)}$, which are the average secure node degree by using only key paths consisting of one proxy in RKP and one logical intermediate node in RKP-DE, and the improvement of $D_s^{RKP-DE(1)}$ over $D_s^{RKP(1)}$ respectively. The expressions for $D_s^{RKP(1)}$ and $D_s^{RKP-DE(1)}$ are the same as the equations (11) and (12) except that we replace P_{couple} and $P_{decouple}$ by $P_{couple}^{(1)}$ and $P_{decouple}^{(1)}$ (derived in [11]) respectively. We can see that network decoupling improves the average secure node degree under all situations, which helps to enhance the performance of random key pre-distribution in terms of connectivity and resilience as demonstrated in the next section. We also observe that the improvement in case of multiple logical intermediate nodes (IM) diminishes for larger D_p. This is because in highly dense network, most physically neighboring nodes are able to establish pair-wise keys. Therefore, the value of D_s^{RKP} is close to that of D_p, and the improvement diminishes.

In the above, we show the derivation of average secure node degree of our RKP-DE protocol. Compared with RKP-DE protocol, our RKP-DEA protocol can achieve even higher average secure node degree. This is because we allow poor nodes to increase their information areas so that they can establish pair-wise keys with more neighbors. We will compare the average secure node degree of RKP-DEA protocol with RKP and RKP-DE protocols via simulation in Section VII.

B. Stretch Factor

In this paper, we define a new metric called $stretch\ factor$ to study protocol communication overhead. Formally, stretch factor is the average number of physical hops on the key path between two secure neighbors. It reflects the communication overhead of a protocol since a message needs to be transmitted/forwarded once for each physical hop during key shares transmission, which dominates the other components in the protocol in terms of communication overhead. We denote the stretch factor for the RKP protocol and RKP-DE protocol as SF^{RKP} and SF^{RKP-DE} respectively and derive them below.

Let us denote $P_h(i)$ and $P_h(i)'$ as the probability that a node can find a logical key path to a physically neighboring node within its information area with minimum logical hop i in RKP-DE protocol and RKP protocol respectively. We further denote α as the average number of physical hops for a logical hop on a key path (except the first logical hop). Then, SF^{RKP} and SF^{RKP-DE} are given by,

$$SF^{RKP} = \sum_{i=1}^{\infty}((1+(i-1)\alpha)\cdot P_h'(i))/\sum_{i=1}^{\infty}(P_h'(i)), \tag{14}$$

$$SF^{RKP-DE} = \sum_{i=1}^{\infty}((1+(i-1)\alpha)\cdot P_h(i))/\sum_{i=1}^{\infty}(P_h(i)). \tag{15}$$

In the above equations, $1+(i-1)\alpha$ denotes the average physical hops of a key path with i logical hops. This is because the first logical hop is between the source node and one of its physical neighbors (resulting in one physical hop). For each of the remaining $i-1$ logical hops, the two nodes of that logical hop are within communication range (one physical hop) with probability 0.5865 [5], and are connected by the source node (two physical hops) with probability $1-0.5865=0.4135$. Therefore, each of the remaining $i-1$ logical hops has average number of physical hops $\alpha = 1\cdot 0.5865 + 2\cdot 0.4135 = 1.4135$. The derivations of $P_h(i)$ and $P_h(i)'$ are given in Appendix.

In Table II, we show the values of SF^{RKP} and SF^{RKP-DE} under various D_p for $K = 10000$ and $k = 50$. We can see that the stretch factor in RKP-DE protocol is only slightly larger than that of the RKP protocol for the same D_p, which is due to dual effects of network decoupling discussed below. With network decoupling, our RKP-DE protocol can construct more key paths than the RKP protocol. So we can partition the set of key paths constructed by RKP-DE protocol (T^*) into two disjoint subsets T_1^* and T_2^*, in which T_1^* denotes the set of key paths constructed by both RKP and RKP-DE protocols, and T_2^* denotes the set of key paths constructed by only RKP-DE protocol. With network decoupling, stretch factor of key paths in T_1^* is decreased. However, stretch factor of key paths in T_2^* is larger than

TABLE II
COMPARISON OF STRETCH FACTOR IN RKP PROTOCOL AND RKP-DE PROTOCOL UNDER DIFFERENT D_p.

D_p	5	10	15	20	25
SF^{RKP}	1.54	1.91	2.01	2.04	2.03
SF^{RKP-DE}	2.14	2.53	2.46	2.37	2.30

that of key paths in T_1^*. The result of the dual effects above is that the stretch factor of key paths in T^* of RKP-DE protocol is slightly larger than that of key paths in T_1^* of RKP protocol. We first explain the reason of decreased stretch factor of key paths in T_1^* by an example. Suppose there exists a key path between a source node and a destination node with multiple logical hops in RKP protocol. Consider a segment (a part of the key path) such that the two end nodes of the segment share keys but are not physical neighbors. In our RKP-DE protocol, we can replace the above segment with multiple physical hops by a single logical hop with only two physical hops [3]. This can help decrease the number of physical hops and the stretch factor. The reason why stretch factor of key paths in T_2^* is larger than that of key paths in T_1^* is that with network decoupling, many key paths not identified in RKP protocol can now be constructed by RKP-DE protocol. These key paths in T_2^* tend to have larger stretch factor.

In the above, we show the derivation of stretch factor of our RKP-DE protocol. Compared with RKP-DE protocol, our RKP-DEA protocol tends to increase stretch factor when increasing the information areas for poor nodes, while it tends to decrease stretch factor when choosing a subset of key paths for rich nodes. The former is due to the fact that in our adaptive local graphs construction, the extra key paths constructed in larger information areas tend to be longer than those constructed in smaller information areas. The latter is because in our adaptive key paths construction, we tend to choose key paths with fewer logical hops based on equation (9), which effectively decreases stretch factor. Overall, the stretch factor of our RKP-DEA protocol depends on the protocol parameters discussed in Section V.

C. Algorithm Complexity

There are four components in both our RKP-DE protocol and RKP-DEA protocol. In the following, we will derive algorithm complexity of each component.

In the common part of local graphs construction in Algorithm 1, complexity is decided by local logical graph construction in lines (11) to (14) of Algorithm 1, which is $\Theta(D_p^2 k)$. Recall D_p is the average physical

[3]For the case where the information area is one hop communication range, all nodes on the key path are within communication range of the source node.

node degree and k is the chain size. For $RKP\text{-}DEA$ protocol, the execution of deciding information area size has a complexity upperbounded by $\Theta((MAX\_INFOAREA^2 D_p)^2 k)$, where $MAX\_INFOAREA$ is the maximum number of hops of information area.

Since dependency checking in Algorithm 3 is used in key paths construction in Algorithms 2 and 4, we need to obtain the complexity of dependency checking first. The complexity of link dependency checking is $\Theta(h_{avg} k_{avg})$, where h_{avg} is the average number of logical hops on a key path, and k_{avg} is the average number of shared keys on a logical hop. The complexity of path dependency checking is $\Theta(p_{avg}^2 h_{avg}^2 k_{avg})$, where p_{avg} is the average number of logical key paths in a logical key path tree.

After obtaining the complexity of dependency checking above, we can obtain complexity of key paths construction in Algorithms 2 and 4. The complexity of logical key path tree construction is $\Theta(p_{avg} h_{avg}^2 k_{avg})$, and that of physical key paths construction is $\Theta(D_p p_{avg}^2 h_{avg}^2 k_{avg})$ for both protocols.

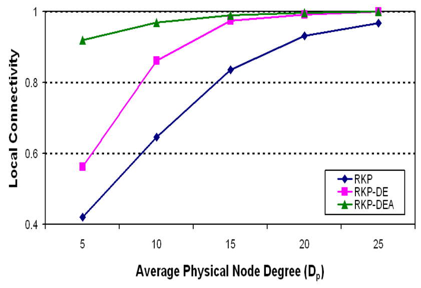
Our key shares transmission has a complexity of $\Theta(D_p p_{avg} h_{avg})$ for both protocols. Overall, the complexity of our $RKP\text{-}DE$ protocol is $\Theta(D_p^2 k + D_p p_{avg}^2 h_{avg}^2 k_{avg})$, and the complexity of our $RKP\text{-}DEA$ protocol is upperbounded by $\Theta((MAX\_INFOAREA^2 D_p)^2 k + D_p p_{avg}^2 h_{avg}^2 k_{avg})$. Based on our analysis and simulation results, the values of p_{avg}, h_{avg}, k_{avg} are small (less than 10) under reasonable parameters. Therefore, our protocols have acceptable complexity for energy constrained sensor nodes.

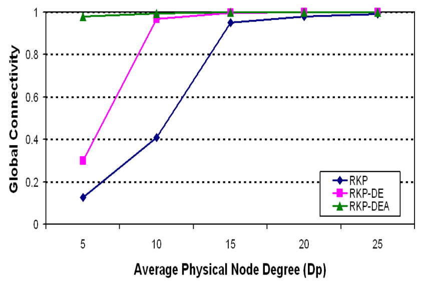
VII. Performance Evaluations

In this section, we report experimental data to demonstrate the performance of our $RKP\text{-}DE$ and $RKP\text{-}DEA$ protocols compared to the traditional RKP protocol under various network and attack parameters. The metrics we study are connectivity (local connectivity and global connectivity) and resilience.

A. Simulation Environment

The sensor network is a square region of size $1000m$ * $1000m$, in which 1000 sensors are deployed uniformly at random. Communication range r is chosen based on the desired average physical node degree D_p. The following are the default values for the parameters unless otherwise specified: information area $A = \pi r^2$, average physical node degree $D_p = 10$, key pool size $K = 10000$, key chain size $k = 50$, required probability that the whole secure network is connected $P_c = 0.99$, and the number of captured nodes $x = 50$. Each simulation is run 100 times with different random seeds, and the data presented is the average of 100 runs.





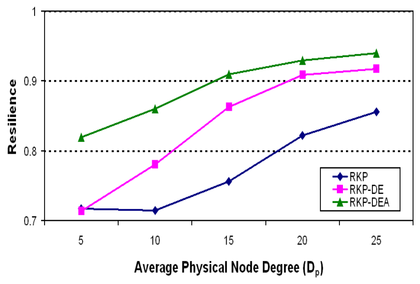


Fig. 7. Sensitivity of local connectivity to average physical node degree D_p

Fig. 8. Sensitivity of global connectivity to average physical node degree D_p

Fig. 9. Sensitivity of resilience to average physical node degree D_p

B. Sensitivity of Connectivity to D_p

1) Local Connectivity: In Fig. 7, we study the sensitivity of local connectivity to average physical node degree D_p. Recall that local connectivity is defined as the probability that two physically neighboring nodes are able to establish a pair-wise key in between. We observe that the local connectivity in RKP-DE and RKP-DEA protocols is consistently higher than that in RKP protocol. The improvement is in fact more significant for non-highly dense networks where $D_p < 20$. In RKP protocol, the consideration of both physical and logical constraints in key path construction limits the availability of key paths between physical neighbors. However, the relaxation of the constraints as a result of network decoupling enables the availability of many more key paths, which greatly enhances the local connectivity. The RKP-DEA protocol further enhances local connectivity by increasing information area for poor nodes.

2) Global Connectivity: The definition of global connectivity here is the percent of nodes in the largest connected component of the secure network (e.g., Fig. 1 (b) or (c)). In Fig. 8, we observe that global connectivity of RKP-DE and RKP-DEA protocols is higher than that of RKP protocol in all situations. The improvement is especially significant in non-highly dense networks. This improvement is a result of the phase transition phenomenon in random graphs [15]. According to this phenomenon, the largest connected component in a random graph with n nodes jumps from $\Theta(\log n)$ to $\Theta(n)$ when the average node degree reaches beyond a certain threshold. With network decoupling in our RKP-DE protocol, such a jump in global connectivity occurs when D_p is around 10 compared to the RKP protocol when D_p is around 15. With the ability to adjust information area, such a jump in RKP-DEA protocol occurs when D_p is even smaller than 5. Another observation is that the global connectivity when $D_p = 5$ in RKP-DEA protocol or $D_p = 10$ in RKP-DE protocol is close to that when $D_p = 20$ in RKP protocol. This demonstrates that one can obtain similar level of global connectivity in our protocols with

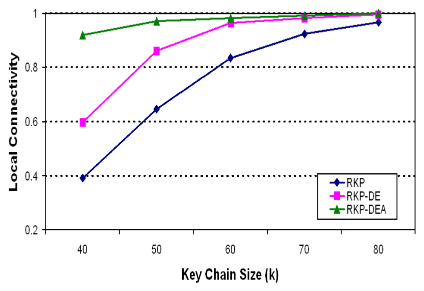


Fig. 10. Sensitivity of local connectivity to key chain size k

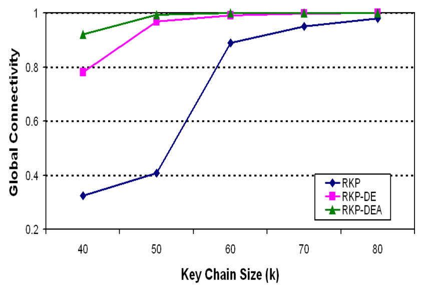


Fig. 11. Sensitivity of global connectivity to key chain size k

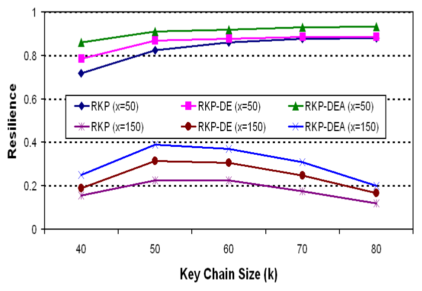


Fig. 12. Sensitivity of resilience to key chain size k and number of captured nodes x

much fewer nodes compared to the basic RKP protocol.

C. *Sensitivity of Resilience to* D_p

In Fig. 9, we study the sensitivity of resilience to D_p. Recall that resilience is defined as the probability that a pair-wise key between two nodes is not compromised given that those two nodes are not captured. We see that resilience is higher in RKP-DE and RKP-DEA compared to that of RKP in general. The improvement is consistent except when the network is very sparse ($D_p = 5$) in RKP-DE protocol. Network decoupling not only increases the number of key paths between two physically neighboring nodes, but also decreases the logical hops of many key paths, both of which help enhance the resilience. When network becomes very sparse, only a single key path can be constructed in most situations in RKP-DE protocol, thus the improvement diminishes. By adjusting information area for poor nodes, our RKP-DEA protocol can achieve significant improvement under sparse networks.

D. *Sensitivity of Connectivity and Resilience to* k *and* x

In Fig. 10, 11 and 12, we study sensitivity of connectivity and resilience to k and x. In Fig. 10 and 11, we see similar pattern in sensitivity of connectivity to k as that to D_p. This is because the increase in k enhances the chance two nodes share keys, which makes the local logical graphs more dense. This can also be achieved by increasing D_p as well. Overall, our RKP-DE and RKP-DEA protocols achieve better performance than RKP protocol, and the performance improvement is especially significant in non-highly dense network. On the other hand, given the same performance requirement, our RKP-DE and RKP-DEA protocols can save storage overhead (proportional to k) up to around 30% and 45% respectively compared with RKP protocol. For example, given $k = 80$ in RKP protocol, our RKP-DE and RKP-DEA protocols can achieve similar performance with k around 60 and 45 respectively.

In Fig. 12, we study sensitivity of resilience to k under different values for number of captured nodes x. We observe that the resilience of RKP-DE and RKP-DEA protocols is better than that of RKP protocol for all cases. When x is relatively small ($x = 50$), the resilience increases with k (for k no more than 250) and the improvement diminishes when k increases. This is because when k increases, the local logical graphs become more dense and more key paths can be constructed. While the increased density improves resilience, it also results in diminishing the performance improvement. On the other hand, when x is relatively large ($x = 150$), the resilience first increases with k, and then begins to decrease when k further increases. This is because when both k and x are relatively large, the attacker is able to disclose a significant percent of the pre-distributed keys thereby compromising many more established pair-wise keys, which degrades the resilience. The threshold value for k, beyond which resilience begins to decrease, will decrease with increase in x. Therefore, the resilience of the case when $x = 50$ will also begin to decrease when k further increases (beyond 250), which is confirmed by our simulation but not shown in Fig. 12. We also notice that the improvement of RKP-DE and RKP-DEA over RKP is more pronounced for larger x (i.e., stronger attacks) further demonstrating the effectiveness of our solutions. The value of x does not impact connectivity, so we do not show the sensitivity of connectivity to x.

VIII. Related Work

The RKP protocol has received wide acceptance in WSNs due to its distributedness, simplicity, energy efficiency and scalability. It has served as a foundation for many other works based on random key pre-distribution [5], [7], [8], [9], [10], [16]. In [5] and [9], the performance of basic RKP protocol is enhanced by constructing multiple key paths for pair-wise key establishment. With multiple key paths, as long as at least one path is uncompromised, the pair-wise key is secure. In [7], [8], [10] and [16], the authors extend the basic RKP protocol by pre-distributing key structures (either polynomials or vectors) instead of keys. When number of captured nodes is small, this protocol has much better resilience compared to the basic RKP protocol. We point out that in the above works, a very high network density (average physical node degree between 20 and 250) is assumed to achieve satisfactory performance.

We now discuss two works that share certain similarities with our network decoupling methodology. In $PIKE$ in [17], node IDs of all nodes form a two dimensional virtual ID space with size $\sqrt{N} \times \sqrt{N}$, where N is the number of nodes in the network. Each node is pre-distributed with a unique pair-wise key with each of the nodes whose node IDs lie in the same row or same column of the virtual ID space. For any two neighboring nodes u and v, if their node IDs lie in the same row or same column, they naturally share a pair-wise key. Otherwise, there are two other nodes in the network that share pre-distributed keys with both u and v, each of which can act as a proxy and form a key path with two logical hops between nodes u and v. Another work is $RKEP$ in [6]. In $RKEP$, a key graph is first constructed based on key sharing among nodes, and then the key graph is used to establish pair-wise keys. Similar to $PIKE$, $RKEP$ also guarantees pair-wise key establishment between any two nodes.

As we can see, the above two works share some similarities with our network decoupling in that non-neighboring nodes can form a logical hop on a key path if they share pre-distributed keys. However, there are major differences between these works and our work. Primarily, the core idea of our network decoupling is to decouple the network *after* node deployment. While in $PIKE$ and $RKEP$, the idea of decoupling is combined with key pre-distribution *before* node deployment, which not only limits the way to distributed keys, but also introduces other limitations discussed below. Second, our network decoupling is more general in that it can be applied to any key pre-distribution scheme, while $PIKE$ and $RKEP$ are two special key pre-distribution schemes. Third, $PIKE$ and $RKEP$ incur much more communication overhead. Both $PIKE$ and $RKEP$ require network wise communication for *each* pair-wise key establishment since the logical intermediate nodes can be deployed arbitrarily far away, while our protocols are localized. The localized nature of our protocols come from the fact that we decouple the network *after* node deployment, thus can limit key path construction within a local area. Protocols requiring network wise communication for each pair-wise key establishment is not scalable for large scale sensor networks with energy constraints. Such problem is inevitable for $PIKE$ and $RKEP$ when they need to guarantee pair-wise key establishment under random node deployment. Fourth, both $PIKE$ and $RKEP$ require a routing structure to be constructed before pair-wise key establishment in order to prevent flooding during the search for key paths. However, without the protection of pair-wise keys, routing structure construction is prone to attacks. Such a dilemma does not exist in our protocols. This is because we decouple the network *after* node deployment, thus our protocols are localized and do not need routing structure. Fifth, $PIKE$ requires large memory usage in large scale network, while our protocols do not have such requirement. In $PIKE$, each node is pre-distributed with $2\sqrt{N} - 1$ keys, which increases with the number of nodes (N) in the network. This is not scalable for large scale resource constrained sensor networks, and this problem is inevitable for $PIKE$ to guarantee pair-wise key establishment.

Orthogonal extensions to the basic RKP protocol are exploiting certain network properties to enhance performance or decrease overhead. Works like [18], [19], [20], [21] [22], [23] and [24] use power control, node mobility, channel diversity, hierarchy and deployment knowledge to enhance performance/ decrease overhead. We point out that our network decoupling is orthogonal to all the works above, and can complement them to achieve further performance improvement and overhead reduction.

IX. Final Remarks

In this paper, we proposed *network decoupling* to separate logical key-sharing relationship from physical neighborhood relationship in random key pre-distributed sensor networks. We designed two new key management protocols (RKP-DE and RKP-DEA) in decoupled sensor networks, and also designed a set of rules for eliminating dependencies among key paths. We conducted detailed analysis and extensive simulations to evaluate our proposed solutions from the perspective of connectivity, resilience and overhead. Our data showed that significant performance improvement can be achieved using our solutions in non-highly dense networks, with only a mild increase in overhead. Our future work consists of practically implementing our proposed solutions on the existing sensor network testbed at OSU [25], [26].

References

[1] W. Diffie and M. E. Hellman, "New directions in cryptography," *IEEE Transactions on Information Theory*, vol. IT-22, no. 6, pp. 644–654, November 1976.

[2] R. L. Rivest, A. Shamir, and L. M. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, February 1978.

[3] B. C. Neuman and T. Tso, "Kerberos: an authentication service for computer networks," *IEEE Communications Magazine*, vol. 32, no. 9, pp. 33–38, September 1994.

[4] L. Eschenauer and V. D. Gligor, "A key-management scheme for distributed sensor networks," in *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS)*, November 2002.

[5] H. Chan, A. Perrig, and D. Song, "Random key predistribution schemes for sensor networks," in *Proceedings of IEEE Symposium on Research in Security and Privacy*, May 2003.

[6] A. Wacker, M. Knoll, T. Heiber, and K. Rothermel, "A new approach for establishing pairwise keys for securing wireless sensor networks," in *Proceedings of the 3rd ACM Conference on Embedded Networked Sensor Systems (Sensys)*, November 2005.

[7] W. Du, J. Deng, Y. S. Han, and P. K. Varshney, "A pairwise key pre-distribution scheme for wireless sensor networks," in *Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS)*, October 2003.

[8] D. Liu and P. Ning, "Establishing pairwise keys in distributed sensor networks," in *Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS)*, October 2003.

[9] S. Zhu, S. Xu, S. Setia, and S. Jajodia, “Establishing pairwise keys for secure communication in ad hoc networks: a probabilistic approach,” in *Proceedings of the 11th IEEE International Conference on Network Protocols (ICNP)*, November 2003.

[10] F. Delgosha and F. Fekri, “Threshold key-establishment in distributed sensor networks using a multivariate scheme,” in *Proceedings of the 25th IEEE Conference on Computer Communications (INFOCOM)*, April 2006.

[11] W. Gu, X. Bai, S. Chellappan, and D. Xuan, “Network decoupling for secure communications in wireless sensor networks,” in *Proceedings of 14th IEEE International Workshop on Quality of Service (IWQoS)*, June 2006.

[12] A. Snoeren and B. Raghavan, “Decoupling policy from mechanism in internet routing,” in *Proceedings of the ACM SIGCOMM Workshop on Hot Topics in Networking (HotNets-II)*, November 2003.

[13] H. Kung and S. Wang, “Tcp trunking: Design, implementation and performance,” in *Proceedings of the 11th IEEE International Conference on Network Protocols (ICNP)*, November, 1999.

[14] D. Niculescu and B. Nath, “Trajectory based forwarding and its applications,” in *Proceedings of the 9th ACM International Conference on Mobile Computing and Networking (MOBICOM)*, September, 2003.

[15] J. Spencer, *The Strange Logic of Random Graphs, Algorithms and Combinatorics 22*, Springer-Verlag, 2000.

[16] F. Delgosha and F. Fekri, “Key predistribution in wireless sensor networks using multivariate polynomials,” in *Proceedings of the 2nd IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON)*, September 2005.

[17] H. Chan and A. Perrig, “Pike: peer intermediaries for key establishment,” in *Proceedings of the 24th IEEE Conference on Computer Communications (INFOCOM)*, March 2005.

[18] J. Hwang and Y. Kim, “Revisiting random key pre-distribution schemes for wireless sensor networks,” in *Proceedings of the 2nd ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN)*, October 2004.

[19] Y. Mao and M. Wu, “Coordinated sensor deployment for improving secure communications and sensing coverage,” in *Proceedings of the 3rd ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN)*, November 2005.

[20] M. Miller and N. Vaidya, “Leveraging channel diversity for key establishment in wireless sensor networks,” in *Proceedings of the 25th IEEE Conference on Computer Communications (INFOCOM)*, April 2006.

[21] P. Traynor, H. Choi, G. Cao, S. Zhu, and T. L. Porta, “Establishing pair-wise keys in heterogeneous sensor networks,” in *Proceedings of the 25th IEEE Conference on Computer Communications (INFOCOM)*, April 2006.

[22] D. Huang, M. Mehta, D. Medhi, and L. Harn, “Location-aware key management scheme for wireless sensor networks,” in *Proceedings of the 2nd ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN)*, October 2004.

[23] W. Du, J. Deng, Y. Han, S. Chen, and P. Varshney, “A key management scheme for wireless sensor networks using deployment knowledge,” in *Proceedings of the 23rd IEEE Conference on Computer Communications (INFOCOM)*, March 2004.

[24] D. Liu, P. Ning, and W. Du, “Group-based key pre-distribution in wireless sensor networks,” in *Proceedings of ACM Workshop on Wireless Security (WiSe)*, September 2005.

[25] E. Ertin, A. Arora, R. Ramnath, and M. Nesterenko, “Kansei: A testbed for sensing at scale,” in *Proceedings of the 4th Symposium on Information Processing in Sensor Networks (IPSN/SPOTS track)*, April 2006.

[26] A. Arora, E. Ertin, R. Ramnath, M. Nesterenko, and W. Leal, “Kansei: A high-fidelity sensing testbed,” *IEEE Internet Computing, special issue on Large-Scale Sensor Networks*, vol. 10, no. 2, pp. 35–47, March/April 2006.

[27] D. Huang, M. Mehta, D. Medhi, and L. Harn, “Modeling pairwise key establishment for random key predistribution in large-scale sensor networks,” Tech. Rep., Dept. of CSEE, University of Missouri-Kansas City, http://conrel.sice.umkc.edu/HRP/modelling pair-wise key establishment schemes-v2.pdf, March 2005.

Appendix

In this section, we will present the derivation of $P_h(i)$ and $P_h(i|A')$ defined in Section III. We will first derive some preliminaries, followed by the derivation of $P_h(i)$ and $P_h(i|A')$ using techniques in [27].

A. Preliminaries

Given the key pool size K and the key chain size k ($k \leq K$), the probability that two nodes share t keys is $P_{key}(t) = \binom{K}{t} \cdot \binom{K-t}{2(k-t)} \cdot \binom{2(k-t)}{k-t} / \binom{K}{k}^2 \quad (0 \leq t \leq k)$.

Each sensor is aware of the key sharing and physical neighborhood information in its *information area* with size A. For ease of exposition, we assume each sensor is aware of its one hop information. Therefore $A = \pi r^2$. Note that our analysis can be applied directly to the situation where multi-hop information is available. Since n sensors are uniformly deployed at random in the network with area S, the average number of nodes in the information area is D_p, which is given by (ignoring boundary effect) $D_p = \frac{A}{S} \cdot n$. The average number of nodes in the overlapped information areas of two physically neighboring nodes is $D'_p = \frac{A'}{S} \cdot n$, where A' is the average size of the overlapped information areas of two physically neighboring nodes. Since the information area is a circle, we have $A' = \frac{(\pi - \frac{3\sqrt{3}}{4})}{\pi} A = 0.5865A$ (as given in [5]).

B. Derivation of $P_h(i)$ and $P_h(i|A')$

We denote $P_h(i)$ as the probability that a node a can find a logical key path to a physical neighbor node b in our RKP-DE protocol within the information area of node a with minimum logical hops i. The expression for $P_h(1)$ is given by,

$$P_h(1) = 1 - P_{key}(0) = 1 - \binom{K}{2k} \cdot \binom{2k}{k} / \binom{K}{k}^2 . \tag{16}$$

In order to analyze $P_h(i)$ ($i > 1$), we divide the nodes in the information area of node a (except nodes a and b) into one of the groups $G(a, j)$ ($j \geq 1$). A node s is in group $G(a, j)$ if node a can find a logical key path from itself to node s within its information area with j logical hops, where j is minimum.

We first analyze $P_h(2)$. The probability that there are n_1 ($1 \leq n_1 \leq D_p - 1$)[4] nodes in $G(a, 1)$, given node b does not share a key with node a, is $\binom{D_p-1}{n_1}(P_h(1))^{n_1}(1 - P_h(1))^{D_p-1-n_1}$. The probability that at least one of these n_1 nodes shares keys with node b is $1 - (1 - P_h(1))^{n_1}$. Hence $P_h(2)$ is,

$$P_h(2) = (1 - P_h(1)) \cdot \sum_{n_1=1}^{D_p-1} \left(\binom{D_p - 1}{n_1} (P_h(1))^{n_1} (1 - P_h(1))^{D_p-1-n_1} \cdot (1 - (1 - P_h(1))^{n_1}) \right). \tag{17}$$

[4]Node b is not in $G(a, 1)$. So n_1 can be $D_p - 1$ at most.

We now analyze $P_h(3)$. The probability that there are n_1 $(1 \leq n_1 \leq D_p - 2)$[5] nodes in $G(a,1)$, given there is no key path between nodes a and b within the communication range of node a with fewer than three logical hops, is $\binom{D_p-1}{n_1} v^{n_1}(1-P_h(1))^{D_p-1-n_1}$. The expression here is different from that in deriving $P_h(2)$ because in this case, nodes in $G(a,1)$ do not share a key with node b. Otherwise, a logical key path with fewer than three logical hops exists. Denoting v as the probability that a node shares keys with node a, but does not share a key with node b, given nodes a and b do not share a key, we have $v = (\binom{K-k}{k} - \binom{K-2k}{k})/\binom{K}{k}$. Denote w as the probability that there is at least one node in $G(a,2)$ given there are n_1 nodes in $G(a,1)$, and at least one of the nodes in $G(a,2)$ shares keys with node b. Thus the minimum number of logical hops of the key path between nodes a and b is three. Then w is given by,

$$w = \sum_{n_2=1}^{D_p-1-n_1} \Big(\binom{D_p-1-n_1}{n_2}\Big(1-(1-P_h(1))^{n_1}\Big)^{n_2}\Big((1-P_h(1))^{n_1}\Big)^{D_p-1-n_1-n_2} \cdot (1-(1-P_h(1))^{n_2})\Big). \quad (18)$$

Finally, the expression for $P_h(3)$ is given by,

$$P_h(3) = (1-P_h(1)) \cdot \sum_{n_1=1}^{D_p-2} \Big(\binom{D_p-1}{n_1} v^{n_1}(1-P_h(1))^{D_p-1-n_1} \cdot w\Big). \quad (19)$$

The derivation of $P_h(l)$ $(l > 3)$ is similar to $P_h(3)$. We denote $y(i)$ $(2 \leq i \leq l-2)$ as the probability that there is at least one node in $G(a,i)$ given n_j $(1 \leq j \leq i-1)$ nodes in $G(a,j)$[6], and denote z as the probability that at least one node is in $G(a,l-1)$ and shares keys with node b. We then have,

$$\begin{aligned} y(i) \;&=\; \sum_{n_i=1}^{D_p-1-\sum_{j=1}^{i-1} n_j-(l-1-i)} \Big(\binom{D_p-1-\sum_{j=1}^{i-1} n_j}{n_i}\Big((1-(1-P_h(1))^{n_{i-1}})(1-P_h(1))\Big)^{n_i} \\ &\qquad \cdot \Big((1-P_h(1))^{n_{i-1}}\Big)^{D_p-1-\sum_{j=1}^{i} n_j}\Big), \qquad (20) \\ z \;&=\; \sum_{n_{l-1}=1}^{D_p-1-\sum_{j=1}^{l-2} n_j} \Big(\binom{D_p-1-\sum_{j=1}^{l-2} n_j}{n_{l-1}}\Big(1-(1-P_h(1))^{n_{l-2}}\Big)^{n_{l-1}}\Big((1-P_h(1))^{n_{l-2}}\Big)^{D_p-1-\sum_{j=1}^{l-1} n_j} \\ &\qquad \cdot (1-(1-P_h(1))^{n_{l-1}})\Big). \qquad (21) \end{aligned}$$

The general form of $P_h(l)$ $(l > 3)$ is given by,

$$P_h(l) = (1-P_h(1)) \cdot \sum_{n_1=1}^{D_p-1-(l-2)} \Big(\binom{D_p-1}{n_1} v^{n_1}(1-P_h(1))^{D_p-1-n_1} \cdot \prod_{i=2}^{l-2} y(i) \cdot z\Big). \quad (22)$$

The expressions of $P_h(i|A')$ are the same as those of $P_h(i)$ except by replacing D_p with D'_p.

[5]Node b is not in $G(a,1)$, and at least one other node is in $G(a,2)$ (thus not in $G(a,1)$). So n_1 can be $D_p - 2$ at most.

[6]None of the nodes in $G(a,i)(1 \leq i \leq l-2)$ shares keys with node b, otherwise a key path with fewer than l logical hops exists.